🖂 26 Av. du Général Charles de Gaulle 92150 SURESNES
+33 (0) 1 46 25 97 42
http://www.t-immersion.com

D'Fusion®

tiShaders Support

User Guide

11/10/12

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

The use of shaders compatible with Direct3D9, OpenGL and OpenGL ES 2 is fully supported by the whole D'Fusion production pipeline - from media export with D'Fusion Exporters, to Mobile, @Home and Pro deployment with D'Fusion players and SDK runtimes.

Up until now, D'Fusion supported Direct3D9 and OpenGL APIs, and shaders could be used but required you to (1) develop your own shaders and (2) manually modify your exported material files in order to account for the use of shaders.

D'Fusion now supports OpenGL ES 2, the only cross-platform mobile 3D API allowing for the use of shaders on Mobile platforms.

The use of shaders reduces the processing burden of the CPU, by involving the GPU in the rendering calculations, and makes them especially interesting for Mobile deployment.

D'Fusion further comes with a set of classical real-time shaders, called **tiShaders**, that will allow you to automatically export your media with shaders compatible with Direct3D9, OpenGL and OpenGL ES 2 directly from Maya and 3ds Max.

1.1 About OpenGL ES 2

OpenGL for Embedded Systems (OpenGL ES) is a subset of the OpenGL 3D graphics API specially designed for embedded systems such as mobile phones, PDAs, and video game consoles.

OpenGL ES is used by PlayStation, Android, Nintendo 3DS, Nokia, Samsung, Symbian and of course by Apple with MacOS and iOS.

The OpenGL ES 2 version eliminates most of the former fixed-function rendering pipeline (FFP) in favor of a programmable pipeline, and brings the power of shaders to embedded systems:

- Enables the use of normal and environment maps on mobile platforms
- Enables the use of reflections and hardware accelerated vertex skinning on 3D model animations

OpenGL ES 2 is widely supported on mobile platforms and, by integrating OpenGL ES 2 into D'Fusion, you can now also use shaders for mobile deployment with D'Fusion Mobile SDKs.

D'Fusion stays compatible with the OpenGL ES 1.X prior versions (no use of shaders).

		OpenGL ES 1	OpenGL ES 2
Fixed pipeline OpenGL functionalities		+	-
Shaders		-	+
	Per-pixel illumination	-	+
	Hardware accelerated vertex skinning	-	+
	Normal Maps	-	+
	Envmaps	-	+

Figure 1: OpenGL ES 1 & OpenGL ES 2 device support

1.2 About tiShaders

tiShaders are shaders embedded in D'Fusion: the D'Fusion Exporter, D'Fusion Studio and the D'Fusion players and runtimes all come with built-in tiShaders.

This way, if you decide to export your media with shaders, the D'Fusion Exporters will then reference the appropriate tiShaders in your exported material files, and D'Fusion Studio and D'Fusion runtimes will automatically handle the rest.

tiShaders have also been written so as to be compatible with Direct3D9, OpenGL and OpenGL ES 2, which makes them platform-independent

An exhaustive list of available tiShaders is provided at the end of this document (section 6.4).

1.3 Advantages of tiShaders

1.3.1 Simplified Workflow

tiShaders truly simplify the whole production pipeline with D'Fusion - from media export to AR scenario authoring and deployment: with tiShaders, you can readily have media with shaders directly from start, and go along the whole production without having to worry about editing the material files manually or explicitly adding specific resources to your scenario.

- **Media export:** the D'Fusion Exporters include the option to export your media with shaders: tiShaders are then automatically referenced by name in the material exported files.
- **AR scenario authoring:** the 3D Viewer allows you to preview your media with tiShaders using one of Direct3D9, OpenGL or OpenGL ES 2 rasterizers.
- **AR scenario deployment:** tiShaders are embedded in the D'Fusion players and SDK runtimes so as to ease deployment: there is no need to explicitly manage them as resources of your scenario.

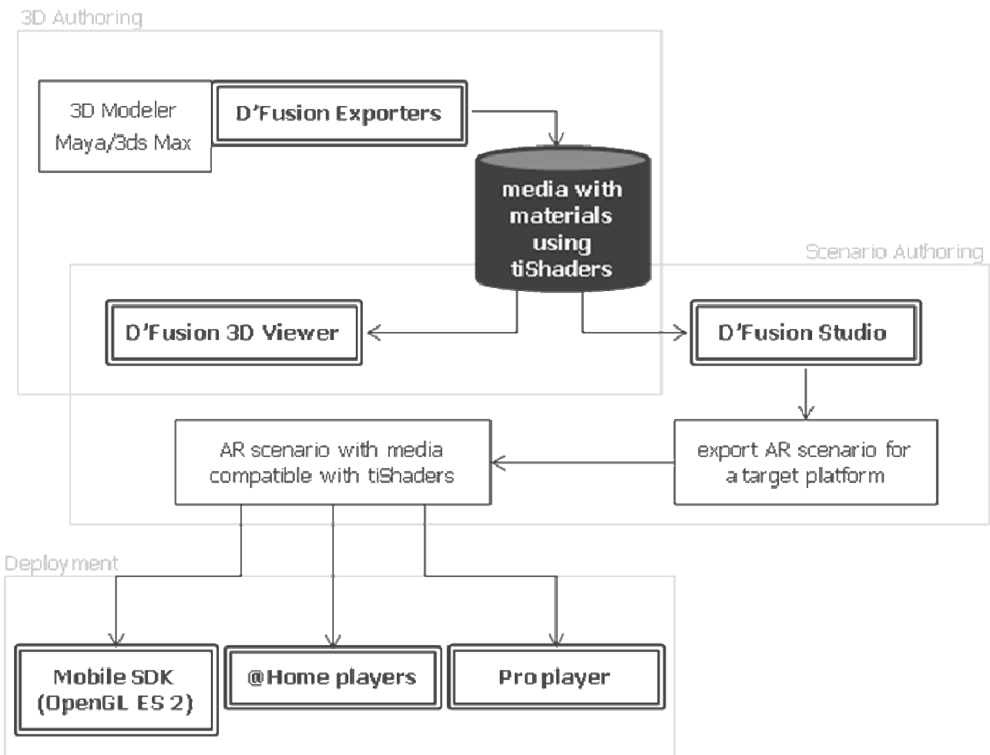


Figure 2: Simplified workflow with tiShaders

D'Fusion also allows you to use your custom shaders. In such case, you will have to manually modify your exported material files to reference your custom shaders, and explicitly include those in your AR scenario resources for authoring and deployment.

1.3.2 Cross-Platform Media Export

tiShaders have the advantage that they are compatible with Direct3D9, OpenGL and OpenGL ES 2 APIs.

Media exported with tiShaders is therefore cross-platform and can be used for @Home, Pro and Mobile deployment.

- Direct3D9 for @Home PC and Pro deployment
- OpenGL for @Home PC / Mac and Pro deployment
- OpenGL ES 2 for Mobile deployment

Cross-platform media further simplifies the programmable production pipeline with D'Fusion.

Note however that AMD graphic cards offer limited support of CG technology. It is thus advised to develop your own custom shaders when deploying on AMD graphic cards. NVIDIA graphic cards on the other hand are recommended when using tiShaders.

DirectX10 compliant graphics card (or higher) are also recommended with tiShaders.

Finally, tiShaders and the use of shaders in general is not recommended for @Home deployment so as to ensure compatibility with the largest possible audience.

1.4 External Documentation

1.4.1 Reference documents

[01] D'Fusion Studio - User Guide, Total Immersion
DFusion Studio - User Guide.pdf

[02] D'Fusion Exporter for Maya - User Manual, Total Immersion
DFusion Exporter for Maya - User Guide.pdf

[03] D'Fusion Exporter for Maya - Modeling Constraints, Total Immersion
DFusion Exporter for Maya - Modeling Constraints.pdf

[04] D'Fusion Exporter for 3dsMax - User manual, Total Immersion
DFusion Exporter for 3dsMax - User Guide.pdf

[05] D'Fusion Exporter for 3dsMax - Modeling Constraints, Total Immersion
DFusion Exporter for 3dsMax - Modeling Constraints.pdf

[06] D'Fusion Mobile - User Guide, Total Immersion
DFusion Mobile - User guide.pdf

[07] D'Fusion @Home – Deployment Guide, Total Immersion
DFusion @Home - Deployment Guide.pdf

1.4.2 Other documents

[08] HLSL Profiles
http://msdn.microsoft.com/en-us/library/bb509626%28v=VS.85%29.aspx

[09] Ogre3D Profiles
http://www.ogre3d.org/docs/manual/manual_18.html#SEC104

2 TISHADERS DESCRIPTION

tiShaders have been written so as to be compatible with Direct3D9, OpenGL and OpenGL ES 2 APIs.
They are embedded in the D'Fusion Exporters, D'Fusion Studio and the D'Fusion players and SDK runtimes, and they can be used independently of the final deployment platform.

2.1 tiShader Effects

tiShaders manage the following effects, described below in more detail:

- Gouraud shading
- Phong shading
- Specular highlights
- Hardware skinning
- Normal maps
- Normal tangents
- Cubic environment maps
- Alpha-blend envmaps
- Diffuse, normal and env textures,

as well as combinations of all of the above (e.g. Gouraud + specular + envmap, Phong + diffuse + normal).

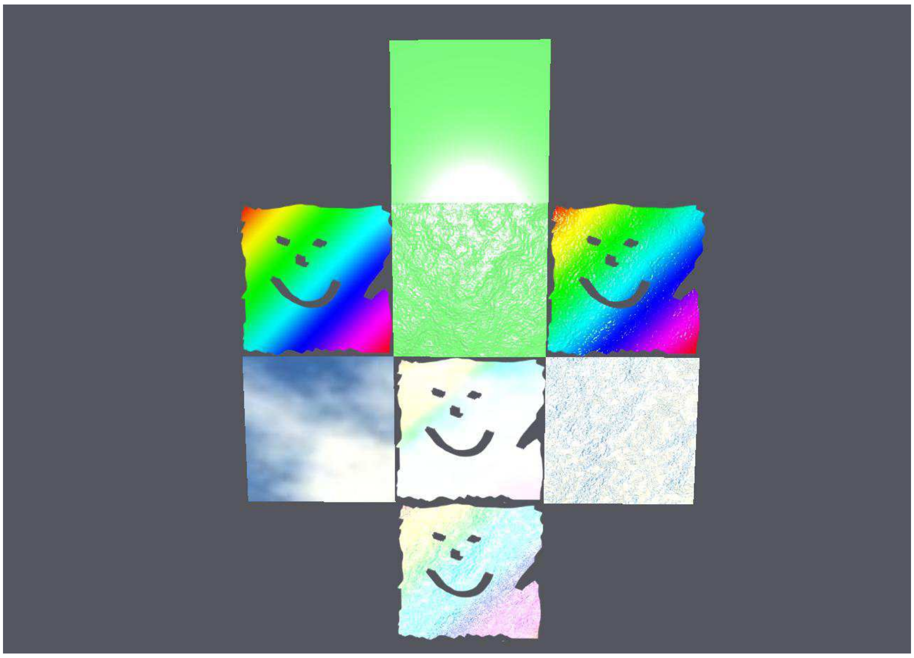


Figure 3: Phong diffuse, normal, envmap, specular and transparency tiShader effects

From top to bottom and left to right, the image above represent the following shader effect combinations:

- Phong
- Phong + diffuse
- Phong + normal
- Phong + diffuse + normal
- Phong + envmap

- Phong + diffuse + envmap
- Phong + normal + envmap
- Phong + diffuse + normal + envmap

A list of available tiShaders and related effects is provided at the end of this document.

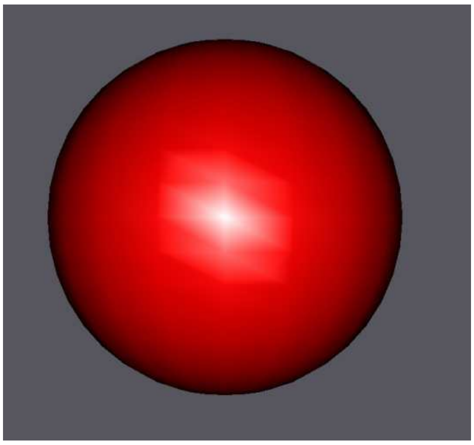
2.1.1 Gouraud and Phong shading

The Gouraud shading algorithm applies an illumination model to each vertex (to calculate the lighting intensity based on the vertex normal), then it linearly interpolates the vertex intensities over the polygon's surface.

Phong shading is similar to Gouraud except that it interpolates the vertex normals (as opposed to vertex intensities).

Effects such as specular highlights for example, can be computed much more precisely with Phong than with Gouraud shading.

Gouraud shading:



Phong shading:

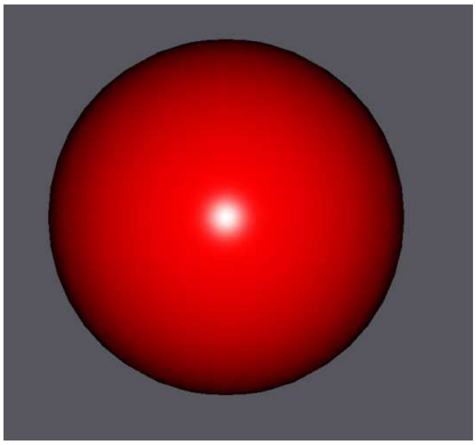


Figure 4: Gouraud vs. Phong shading
With Gouraud, lighting is per vertex,
pixel is LERP of three neighboring vertices

Gouraud shaders are equivalent to OpenGL ES 1 in terms of final render, with the advantage that envmaps can be used for Mobile deployment with OpenGL ES 2 shaders.

Gouraud should be preferred for Mobile deployment since it requires less computation (although better performance also means loss of precision per vertex).

2.1.2 Hardware Skinning

tiShaders allow the use of hardware skinning for your skeleton animations.

With hardware skinning, the skinning is handled on the GPU by a vertex shader, as opposed to software skinning, which is handled on the CPU (Ogre3D's native). GPU computations can be particularly interesting for devices with low-end CPUs such as Mobile devices. The use of hardware skinning shaders with OpenGL ES 2 is therefore highly recommended.

Hardware skinning further eliminates some drawbacks encountered with software skinning. In software skinning, vertices are animated on the CPU and sent back to the graphics card for each frame, which implies a high cost in terms of bandwidth consumption. Secondly, only vertex positions and normals are animated, without transforming the tangents, which can lead to unexpected results with normal mapping.

Performing the skinning animation in the vertex shader solves these drawbacks. Nevertheless, a limitation is placed on the amount of data that can be sent to the shader. More information can be found in section 3.1.2.

2.1.3 Diffuse Maps

A diffuse map is a texture used to define the main color of a surface. It can be a 3 (RGB) or 4 (RGBA) component texture.

Diffuse maps can be combined with normal maps, env maps and specular effects.

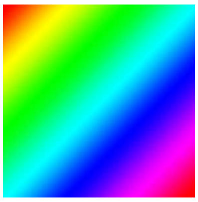


Figure 5: Diffuse map

2.1.4 Normal Maps

Normal maps are used to store normal vectors, which are encoded using three channel colors.

The data contained in normal maps can be expressed in two different coordinate systems: object or tangent space. Tangent space has been preferred for tiShaders, since it allows for mesh skinning transformations.

tiShaders can also build vertex tangents, useful when vector normals appear inverted (as in the example shown in section 3.1.4).

Normal maps always use Phong shading.

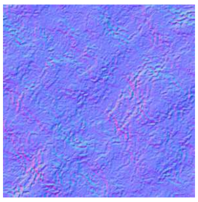


Figure 6: Normal map

2.1.5 Envmaps

Environmental maps (envmaps) are used to fake the reflection of the object's surrounding environment on the object's surface.

tiShaders can further take into account the transparency of the envmap independently of the object's transparency (see alpha-blend below).

Figure 7: Envmap

2.1.1 Envmap Alpha Blend

The envmap alpha-blend takes the transparency of the envmap to be that of the object (the envmap's alpha *blends* with that of the object).

tiShaders can disable the alpha blend of envmaps so as to consider the transparency of the envmap and the transparency of the object as two different attributes.

Disabling the alpha blend of an envmap can be useful for very transparent objects having a very reflective envmap, like a pair of glasses.

2.1.2 Specular Highlights

tiShaders can also take into account a specular component which modulates specular highlights.

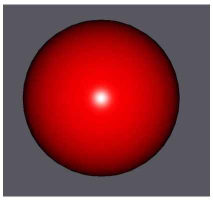


Figure 8: Phong with specular highlight

2.2 Vertex and Fragment tiShaders

Shaders usually work in pairs: a vertex shader and a fragment shader.

tiShaders also work in pairs: a **vertex tiShader** and a **fragment tiShader**.

Each vertex/fragment shader has a specific function:

- A vertex shader receives a vertex, and must provide the graphics card with a screen position so it can be drawn
- A fragment shader is called for each pixel, and its purpose is to output the color of the pixel (RGBA)

They both have a pre-defined goal, established by the graphics card (hardware skinning effects for example can only be computed through vertex shaders), but the shader's code implementation is left to the user.

Through the programmable pipeline, data can be sent and interpolated from the vertex shader to fragment shader (position, normals, UVs).

In the advanced user section 6.4 below, you will find the list of all available tiShaders and the effects covered by each specific shader.

3 MEDIA WITH SHADERS

You can export media with shaders directly from Maya and 3ds Max using built-in tiShaders: the D'Fusion Exporters do all the necessary calculations to automatically reference the appropriate tiShaders in the material exported files, for all targets and platforms.

Media exported with tiShaders is cross-platform. General real-time media limitations specific to each platform are thoroughly explained in documentation [03] and [05].

The following section describes media limitations specific to tiShaders.

Section 3.2 will explain you how to export with tiShaders from Maya and 3ds Max using the D'Fusion Exporters.

You can also consider developing your own custom shaders and edit the exported material files manually to reference your shaders.

3.1 Media Limitations with tiShaders

3.1.1 General real-time limitations

As a reminder, the following table reviews per-platform general real-time limitations.

	@Home	Pro	Mobile
Modeling: Number of polygons & Number of animations	Around 150.000 polygons depending on the number of animations	Depending on the PC you used, up to 800.000 without animations	Around 10.000 polygons, depends on the number of animations
Exporter: Targeted Hardware Profile	Public	Pro	Public/Custom
Use of Shaders	Yes (but not recommended)	Yes	Yes (with OpenGL ES 2)
Environment Maps	Yes	Yes	Yes (with shaders only)
Texture size	512x512 pixels	Depends on the screen resolution. For example, for a screen in 1280x720, a texture in 1024x1024 pixels is enough	256x256 pixels
Texture type	.jpg and .png	.bmp, .jpg, .png, .tga	.jpg and .png

Figure 9: General real-time limitations

Please visit the 3D authoring documentation [02], [03], [04] and [05] for more detailed information.

3.1.2 Hardware Skinning Limitations

With Hardware Skinning, a limitation is placed on the amount of data that can be sent to the shader. The number of bones making up the skeleton (NB_MESH_BONES) and the number of bones per vertex (NB_VERTEX_BONES) is thus limited.

The following table specifies both hardware and software skinning limitations with Ogre3D:

	@Home	Pro	Mobile	
	Direct3D9/OpenGL		OpenGL ES 1	OpenGL ES 2
Hardware skinning	70 bones		not supported	39 bones
Hardware / Software skinning	Max 4 bones per vertex			

Figure 10: Hardware and Software skinning limitations

To overcome the bone limitation, you can assign two different materials to the same mesh, which will then be split at the export.

3.1.3 Light Limitations

The number of lights and the type of light used in the scene (point/spot/directional) has a direct influence on the shader's computational cost.

The following limitations have been kept for tiShaders:

	@Home	Pro	Mobile
	Direct3D9 / OpenGL with shaders		OpenGL ES2
# of lights	3 lights max		1 light max
light type	pointlight / spotlight		pointlight

Figure 11: Light limitations with shaders

3.1.4 Normal Maps Limitations

Normal maps are only available with Phong high quality export (Phong HQ).

Gouraud low quality export does not support normal maps.

D'Fusion supports normal maps defined in tangent space. Object space normal maps are not supported.

Supported: Tangent space normal maps	**Not supported: Object space normal maps**

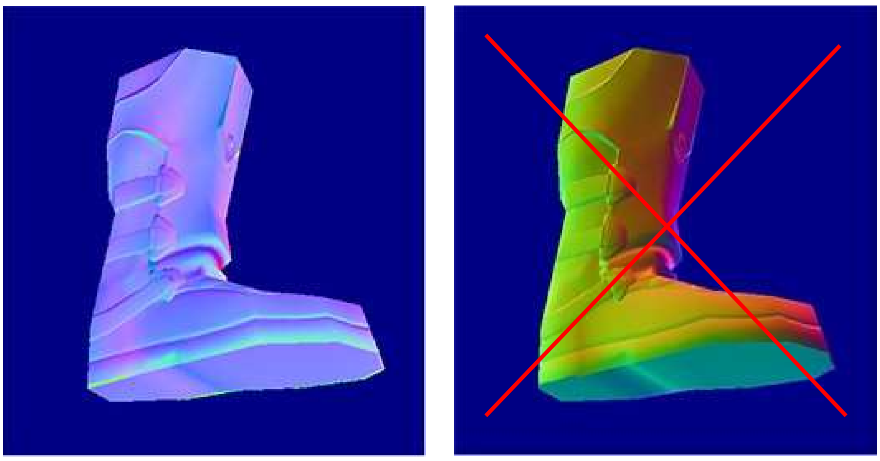


Figure 12: Tangent vs. object space normal maps per platform

To use normal maps, the "Custom" target hardware profile with tangent option enabled must be checked at Export.

Bump maps are not supported.

If tangent vectors are to be inverted (for example, when the tangents on a surface are found to be discontinuous), you can use the export option "Parity bit (Normal Map)". This option requires extra computing calculations and should not be used unless necessary.

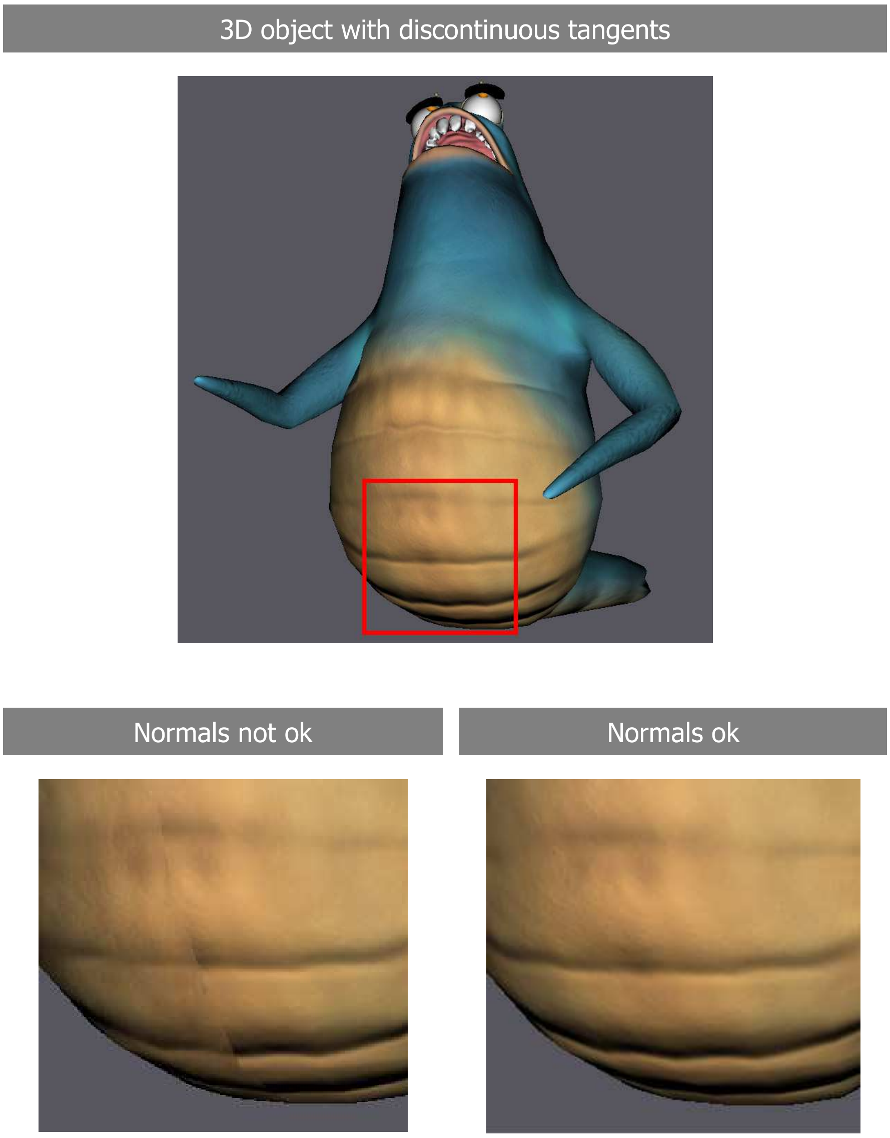


Figure 13: Example of Parity Bit option usage

3.1.5 Environment Maps Limitations

When exporting with shaders, only cubic envmaps (provided as six separate images) are supported.

Supported: **Cubic Envmaps** ***6 separate files***	**Not supported:** **Spherical Envmaps, Planar Envmaps**

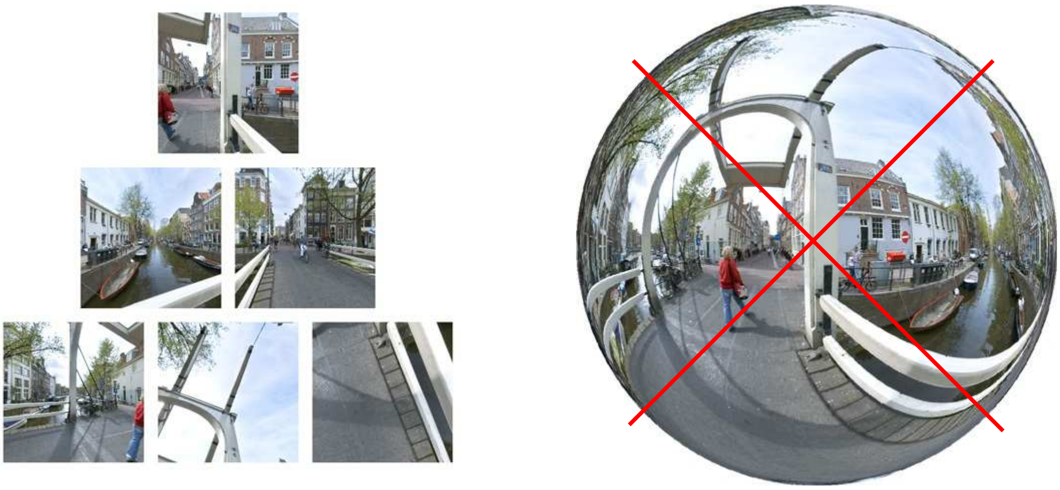


Figure 14: Supported envmap types

This means that you need to set your envmaps to cubic in your Maya/3ds Max scene if you want to use shaders.

You can still export your media without shaders (compatible with Direct3D9, OpenGL and OpenGL ES 1) if you want to use non-cubic env maps (although envmaps are not supported by OpenGL ES 1).

<table>
<tr><th>@ Home</th><th>Pro</th><th colspan="2">Mobile</th></tr>
<tr><th colspan="2">Direct3D9 / OpenGL</th><th>OpenGL ES 1</th><th>OpenGL ES 2</th></tr>
<tr><td colspan="2">cubic envmaps only with shaders</td><td rowspan="2">envamps not supported (no shader support)</td><td rowspan="2">cubic envmaps only (with shaders)</td></tr>
<tr><td colspan="2">all envmaps supported without shaders</td></tr>
</table>

Figure 15: Envmaps per platform

More information concerning shaders and envmaps (like the alpha-blend property) is available from the Exporters documentation [02] and [04].

3.1.6 Material Attributes

Not all material attributes are supported when exporting with tiShaders.

All technique attributes are supported, but some pass and texture attributes are not taken into account or are taken into account in a way different from the default Ogre3D (without shaders).

We recommend reading through section 6.2, which provides a complete description of supported material attributes, if you are planning to develop your own custom shaders.

3.2 Export with tiShaders

The D'Fusion Exporters allow you to export your media with tiShaders directly from Maya and 3dsMax.

This section explains you the different shader export options available with the D'Fusion Exporters.

3.2.1 D'Fusion Exporter window

The "Parameters" section of the D'Fusion Exporter for Maya provides you with several options to export with tiShaders.

Figure 16: D'Fusion Exporter for Maya "Parameters"

The same options are available from the "Shaders" section of the D'Fusion Exporter for 3dsMax:

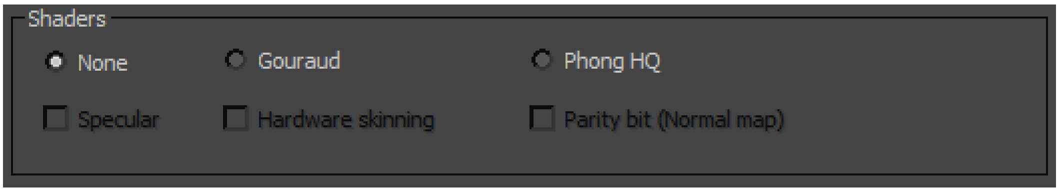


Figure 17: D'Fusion Exporter for 3dsMax "Shaders"

In accordance with the options you choose, the exporters will make all the necessary calculations to reference the appropriate tiShaders in your exported material files.

3.2.2 Shader Export Options

You can choose between two main types of exports:

- **Gouraud**, which uses standard quality shaders
- **Phong**, which uses high quality shaders

The choice of low/high quality should depend on the hardware you are targeting and the frame rate you expect. **The use of high quality shaders for example may result on a lower frame rate on some hardware such as Mobile devices**.

For Mobile deployment thus, a low quality export should be preferred.

High quality export is recommended for Pro deployment, for example, especially when final hardware conditions can be tested before final deployment.

You can visit section 6.5 for a comparative list of tiShaders performances per device.

With Gouraud/Phong export, the following effects are available:

- **Specular:** If checked, the specular highlights will be computed by the fragment program. This option may be consuming in terms of performances.
- **Hardware skinning:** If checked, the animation of skinned objects is carried out by the vertex program.
- **Parity bit:** This is related to normal maps that may require the use of the tangent W component.

Normal maps are not compatible with Gouraud (low quality) export, so the "Parity bit" option is not available with Gouraud export.

All other shader effects are available in both modes, tangents are not needed and the public profile export mode suffices (instead of custom with tangent).

Low Quality	High Quality
Gouraud	Phong
Hardware skinning	Hardware skinning
Specular	Specular
Envmap	Envmap
	Normal maps
	Parity bit

Figure 18: High quality vs. low quality export

3.2.3 Per-Platform Recommendations

Shaders are not supported by all APIs, and some graphic cards do not support shaders.

Because of the latter one, we advise not to use shaders for @Home deployment.

The following table reviews the options recommended to use at export depending on your target platform and API.

Expected Effect	Target Platform	Required Export Options
Hardware Skinning	ALL	Shaders enabled Hardware Skinning enabled
Environment Maps	Mobile	Shaders Enabled Use cubic envmaps only
	@Home	Export without shaders recommended Cubic or spherical
Normal Maps	Mobile	Custom export with Build tangents option Shaders Enabled Phong HQ

Figure 19: Per-platform shader export recommendations

The following table reviews export options required to use effects enabled (or disabled) by the use of shaders depending on the target platform:

	@Home	Pro	Mobile	
	Direct3D9/OpenGL		OpenGL ES 1	OpenGL ES 2
Shaders usage	not recommended	highly recommended	not supported	compulsory
Shaders export option	none	Phong (HQ) recommended	none	Gouraud (LQ) recommended

Figure 20: Shader effects and required export options

3.3 Particles and Overlays

Particles and Overlays material files are not generated by the D'Fusion Exporters, and so the use of tiShaders is not automatic. As a consequence, particles and overlay material files are to be modified manually if you want to use shaders (even if built-in tiShaders are used).

You will especially have to modify the particle and overlay material files when deploying for Mobile platforms with OpenGL ES 2 API. This is because OpenGL ES 2 requires all your media to use shaders (particles and overlays included).

With @Home and Pro deployment, on the other hand, you can mix media with and without shaders (see table

		OpenGL ES 1	OpenGL ES 2
Fixed pipeline OpenGL functionalities		+	-
Shaders		-	+
	Per-pixel illumination	-	+
	Hardware accelerated vertex skinning	-	+
	Normal Maps	-	+
	Envmaps	-	+

Figure 1 for details), which means that you can export your media with/without shaders and use your particles and overlays as generated by the standard Ogre3D tools (i.e. without shaders).

D'Fusion provides tiShaders specific for particles and overlays. How to reference the appropriate tiShaders in your particles' and overlays' material files is explained in section 6.3.

3.4 Use of Custom Shaders

Exported material files can be manually modified to account for the use of custom shaders.
Custom shaders set in Maya/3dsMax will not be taken into account by the D'Fusion Exporters.

If you want to use your custom shaders:

- Use standard materials (Bling, Lambert) in Maya/3dsMax;
- Export your media with shaders (tiShaders) by choosing between Gouraud (LQ) / Phong (HQ) and setting the Specular/Hardware Skinning/Parity Bit options;
- Manually modify the exported material files to reference your custom shaders.

You will also need to add your custom shaders to your scenario as resources of your project before deployment. Refer to section 4.5 and 4.6 for more information.

4 SCENARIO AUTHORING WITH SHADERS

You can preview your media in D'Fusion Studio and the 3D Viewer using one of the supported Direct3D9, OpenGL and OpenGL ES 2 APIs.

Media exported with tiShaders can be previewed with no additional considerations because tiShaders are embedded in D'Fusion Studio and the 3D Viewer.

Custom shaders on the other hand (i.e. the ones you develop) are to be added to your project resources, and need to be included for export with your project before deployment.

4.1 Mixing Media with and without Shaders

Depending on the renderer, you may (or may not) be able to mix media with and without shaders.

Direct 3D9 and OpenGL for example, allow you to mix media with and without shaders.

OpenGL ES 2, on the other hand, requires all your media to use shaders, and OpenGL ES 1 does not support shaders at all (so all your media must be shader-free).

<table>
<tr><th></th><th colspan="2">Personal Computer</th><th colspan="2">Mobile</th></tr>
<tr><th></th><th>Windows
Direct3D / OpenGL</th><th>MacOSX
OpenGL</th><th>android</th><th>iPhone</th></tr>
<tr><th></th><th></th><th></th><th colspan="2">OpenGL ES 1 / OpenGL ES 2</th></tr>
<tr><td rowspan="2">mix media with and without shaders</td><td rowspan="2">ok</td><td rowspan="2">ok</td><td>NOT ok</td><td>NOT ok</td></tr>
<tr><td colspan="2">OpenGL ES 1: NO shaders
OpenGL ES 2: ALL shaders</td></tr>
</table>

Figure 21: Mixing media with and without shaders limitations

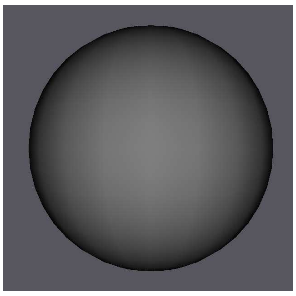
4.2 Preview with Direct3D9, OpenGL, OpenGL ES 2

The 3D Viewer and D'Fusion Studio allows you to choose amongst the Direct3D9, OpenGL and OpenGL ES 2 rasterizers to preview your media while authoring.

Changing the rasterizer can be useful if you wish to simulate the final render of your application while authoring your scenario.

The OpenGL ES 2 rasterizer can also be used to detect media without shaders, highlighted in blinking red by D'Fusion when the OpenGL ES 2 API is chosen. This media should be reviewed to account for the use of shaders, especially if the targeted runtime API is OpenGL ES 2 (all media must use shaders with OpenGL ES 2).

Typical render for a sphere:



Typical render for a sphere with no shaders when the OpenGL ES 2 is chosen for preview:

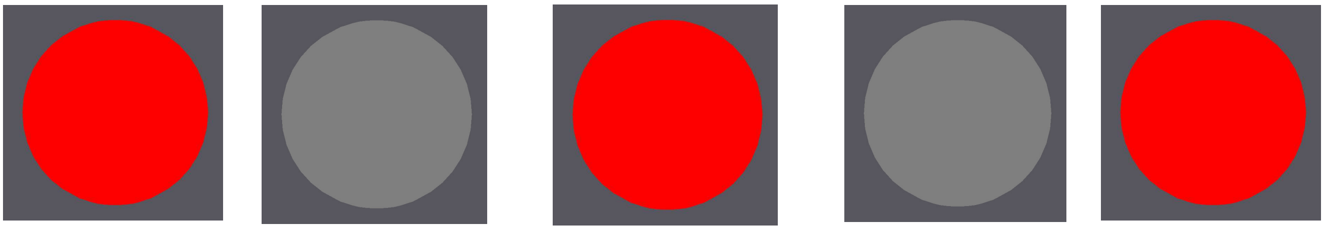


Figure 22: Media without shaders appears in blinking red

Warning: We recommended having a DirectX10 compliant graphics card (or higher) when tiShaders are used. With Direct3D9 and OpenGL (not OpenGL ES 2), it is also recommended to use NVIDIA graphic cards since AMD graphics cards offer limited support of CG technology.

4.2.1 Choosing the rasterizer in D'Fusion

To choose the rasterizer to use while in D'Fusion Studio, go to application preferences: *Menu bar > Application > Preferences > General > Rasterizer*

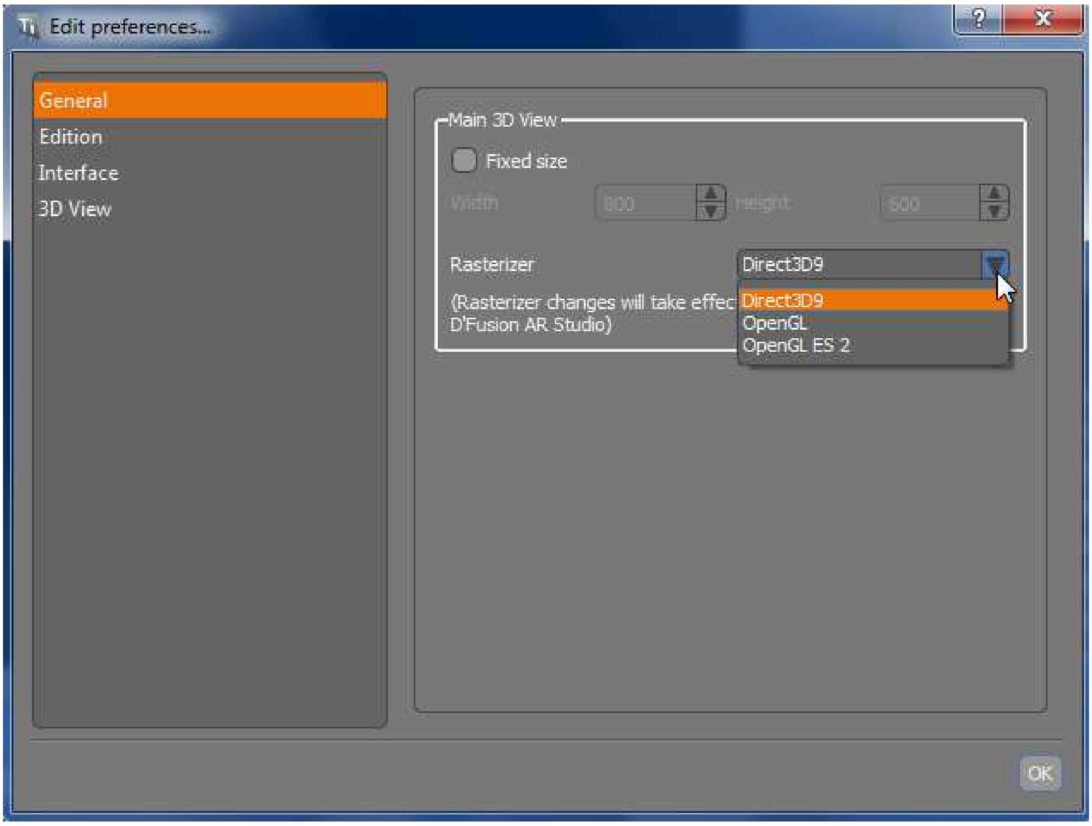


Figure 23: Choosing the rasterizer for D'Fusion Studio

You must restart D'Fusion Studio for changes to take effect.

Note that the rasterizer you choose here is different from the rasterizer that will be used at runtime (which you can specify from the Project settings, as described in section 4.3). The one chosen here will be used for preview only while in D'Fusion Studio (or while in the 3D Viewer as described below).

4.2.2 Choosing the rasterizer in the 3D Viewer

In the 3D Viewer, you can change the Rasterizer to preview your media from the Settings window: *Menu bar > Window > Settings*

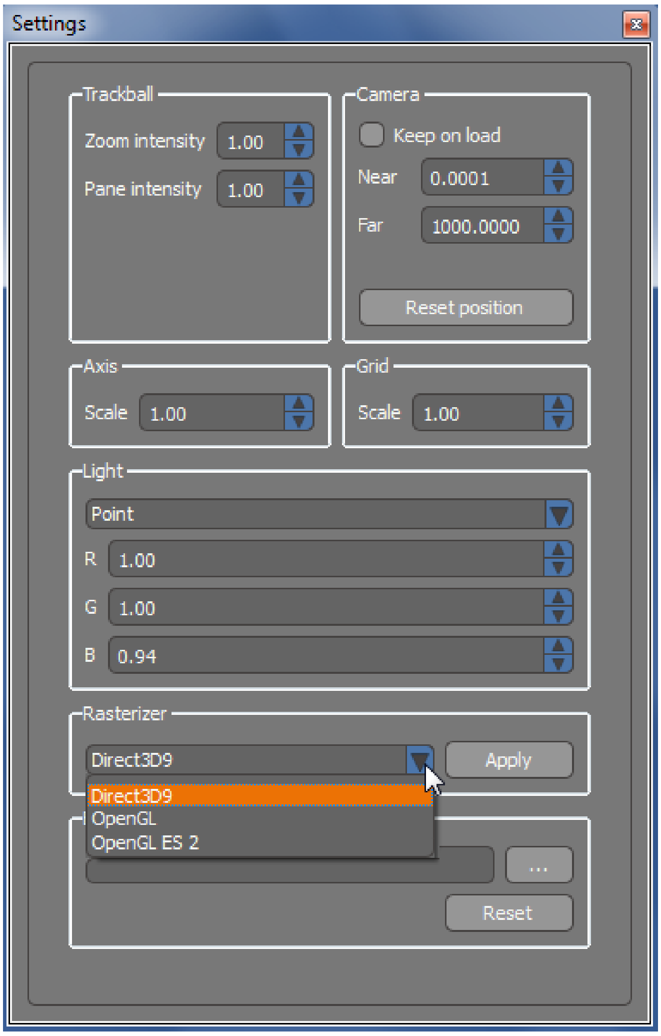


Figure 24: Choosing the rasterizer for the 3D Viewer

When in D'Fusion Studio, the 3D Viewer will use the rasterizer specified in the application preferences.

4.2.3 Troubleshooting

If you change the D'Fusion Studio/3D Viewer rasterizer to one not supported by your graphics card (i.e. the one used for authoring) D'Fusion Studio may fail to launch when restarting the application. To workaround this issue:

- Close D'Fusion Studio/3D Viewer
- Open the D'Fusion Studio/3D Viewer preferences file (%appdata%\Total Immersion\DFusionStudio.ini / %appdata%\Total Immersion\DFusion3DViewer.ini)
- Delete the line containing "dfsMainView3DRasterizer=XXX" (XXX being the not-supported API, e.g. OpenGL ES 2)

Next time you launch D'Fusion Studio/3D Viewer, the default Direct3D9 rasterizer will be used.

4.3 Project Default Runtime Rasterizer

D'Fusion Studio also allows you to choose the default rasterizer to use for your project at runtime.

This option is available from the Project Settings window: *Menu bar > Project > Settings > Rendering > Rasterizer.*

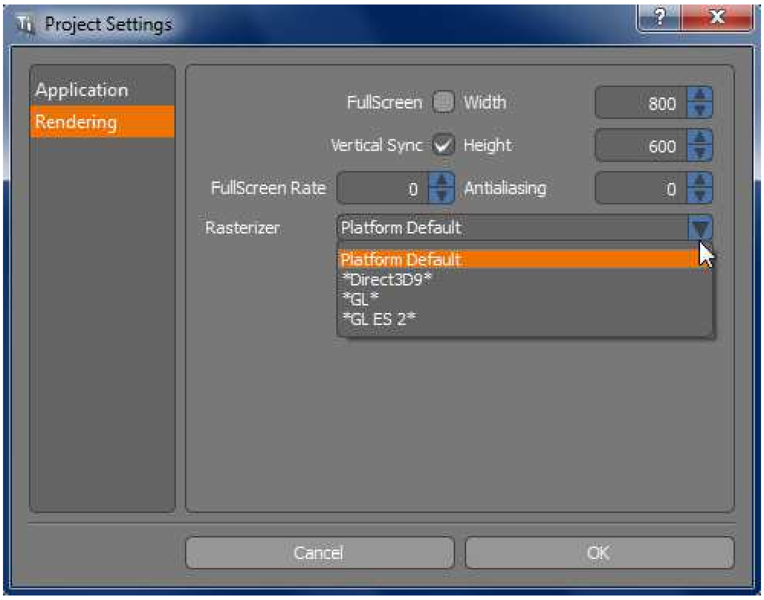


Figure 25: Project's default runtime rasterizer

"Platform Default" (recommended) uses Direct3D9/OpenGL for @Home/Pro, and OpenGL ES 2 for Mobile.

Please note the OpenGL ES 2 rasterizer is not supported for the Mac OS X platforms.

4.4 3D Geometry Info

The 3D Viewer "Geometry" window allows you to check on specific media limitations, such as the maximum total number of bones and the maximum number of bones per vertex.

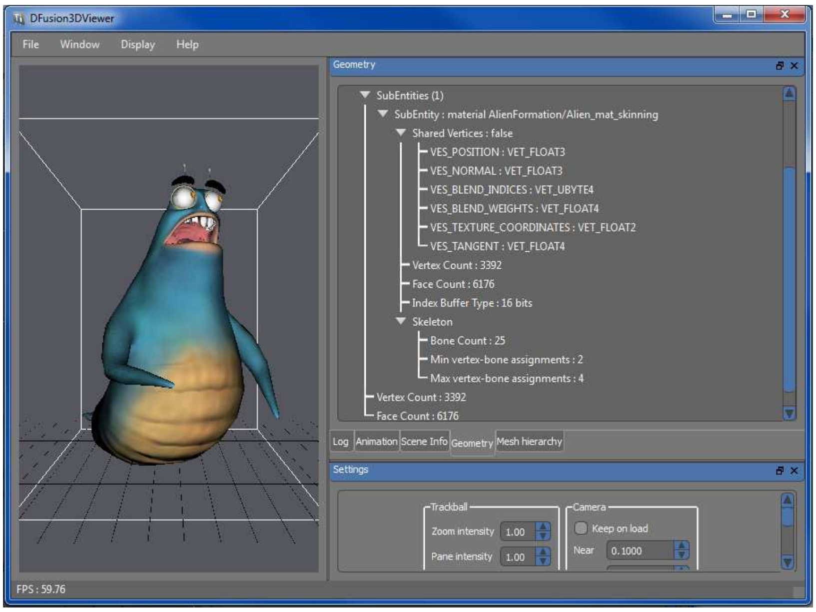


Figure 26: 3D Viewer's Geometry info

Tip: Do no forget to right click on the media you want to check!

Geometry information is also available from within D'Fusion Studio's "3D Viewer" tool. In general, all D'Fusion 3D Viewer features are also available from within D'Fusion Studio.

4.5 Use of Custom Shaders

tiShaders are embedded in D'Fusion Studio, which means that you do not need to add them to your project resources when authoring in order to preview your media.

However, if you developed your own shaders, you need to explicitly add them to your project resources (and include them for export as explained in the next section).

We further recommend saving your custom shaders next to your geometry to preview your media in the D'Fusion 3D Viewer.

To add your custom shaders to your project, you can proceed like for any other type of resource (3D, scripts, etc):

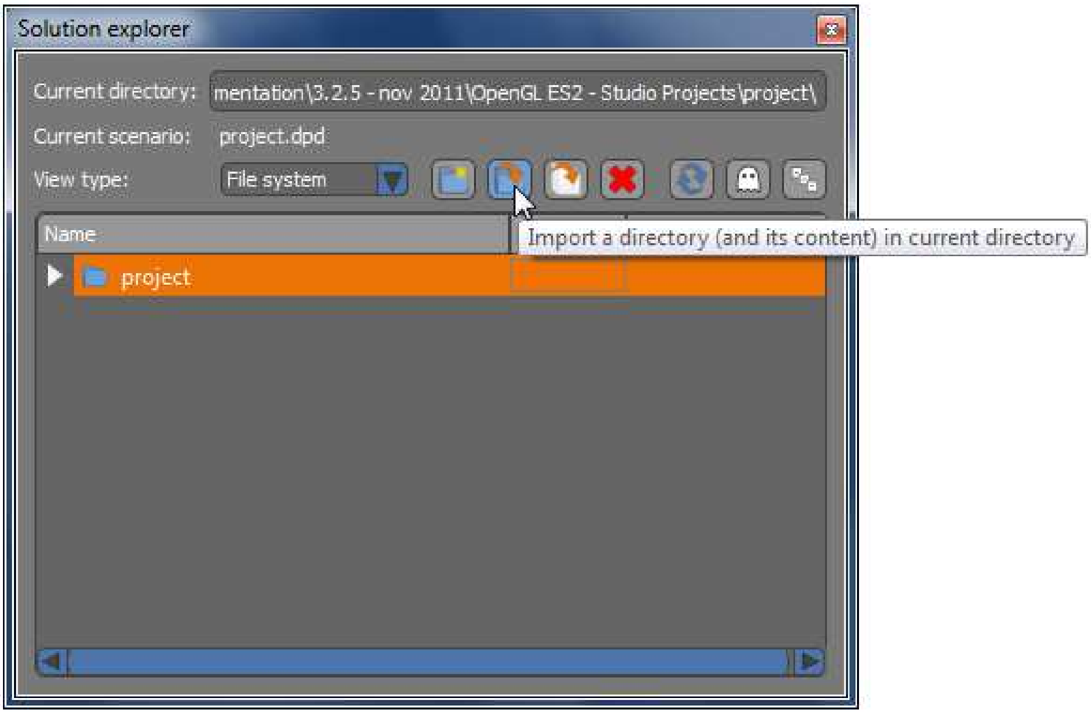


Figure 27: Import custom shader resources

Custom shaders are to be placed next to your texture and .scene files if using the 3D Viewer for previsualization.

4.6 Export with Custom Shaders

tiShaders are embedded in D'Fusion players and SDK runtimes, which means that you do not need to export them with your AR scenario before final deployment.

However, if you developed your own shaders, you need to add them to your project resources when authoring (as explained above) and include them for export when exporting your project from D'Fusion Studio (as explained here).

To export your custom shaders with your project, right-click on the directory/files containing your custom shaders and select "Include in export list":

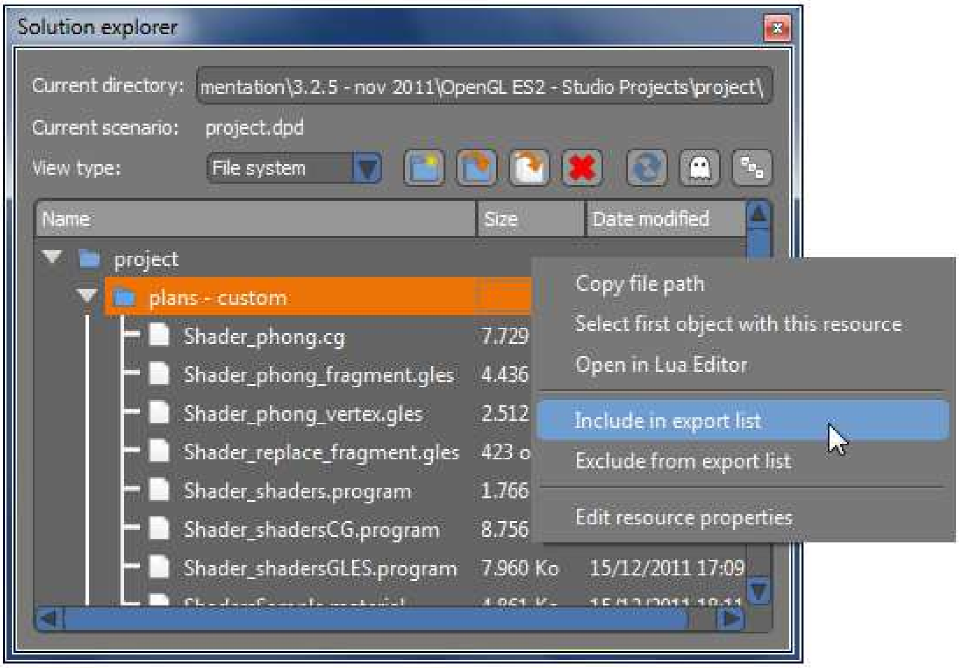


Figure 28: Include custom shaders in the project export

5 Deployment with Shaders

tiShaders are embedded in the D'Fusion players and SDK runtimes to ease your deployment. If you are using custom shaders, you must make sure that they are properly exported with your project before you proceed to final deployment (as explained in sections 4.5 and 4.6).

The following sections provide further information on the use of shaders depending on the targeted deployment platform.

We assume you have some knowledge of Mobile, @Home and Pro deployment, and that you have read the relevant documentation.

5.1 Mobile Deployment

D'Fusion supports OpenGL ES 1 and OpenGL ES 2 for mobile deployment.

The use of the OpenGL ES 2 API is highly recommended for Mobile deployment, especially since it is widely supported on AR-adapted devices while offering greater render possibilities.

		OpenGL ES 1	OpenGL ES 2
Fixed pipeline OpenGL functionalities		+	-
Shaders		-	+
	Per-pixel illumination	-	+
	Hardware accelerated vertex skinning	-	+
	Normal Maps	-	+
	Envmaps	-	+

Figure 29: OpenGL ES 1 vs. OpenGL ES 2 features

5.1.1 Media Recommendations with OpenGL ES 2

If you have decided to deploy your mobile application using the OpenGL ES 2 API (recommended), the use of **shaders is compulsory** on each and every object of your scene.

It is strongly recommended to (re)export your media with tiShaders if you are targeting the OpenGL ES 2 API, even if you wish to reuse media exported for the OpenGL ES 1.X API, so as to make sure that all your media use shaders.

If you are using **Particles and Overlays** and you are targeting the OpenGL ES 2 API, do not forget to modify the particle and overlay material files manually too in order to account for the use of shaders (tiShaders or your own custom shaders).

Section 6.3 provides more details on how to add the use of shaders to your particle and overlay material files.

The Mobile deployment documentation [6] also provides more specific information on media requirements regarding Mobile platforms.

5.1.2 Component Initialization

The D'Fusion Mobile tiComponent needs to be initialized with the appropriate 3D API (OpenGL ES 1 or OpenGL ES 2).

iPhone and Android devices require a different initialization. You should refer to the Mobile deployment documentation [6] for specific information on the tiComponent description and initialization.

5.2 @Home Deployment

D'Fusion players embed tiShaders for @Home development.

However, because not all graphic cards support shaders, and @Home is designed to target the widest possible audience, the use of shaders (tiShaders or custom shaders) is **not recommended** for @Home deployment. The use of standard materials without shaders should be preferred.

It is therefore recommended to **export your media without shaders** so as to ensure that you target the largest possible audience.

5.3 Pro Deployment

Pro runtimes support Direct3D9 and OpenGL renderers.

This means that you can mix media with and without shaders for your Pro applications.

Shaders should be however tested on the targeted hardware to avoid encountering problems with the graphics card drivers.

AMD graphics cards for example offer limited supported of CG technology, and it is advised to develop your own custom shaders and avoid using tiShaders when deploying on AMD graphic cards. NVIDIA graphic cards are recommended when using tiShaders.

6 ADVANCED USERS

6.1 Custom Shader Development

If you wish to develop your own shaders, you will have to:

(1) Write your shaders using the appropriate shading language;
(2) Define a program file (".program") so that Ogre knows how to read them;
(3) Modify your material files manually so as to reference your custom shaders.

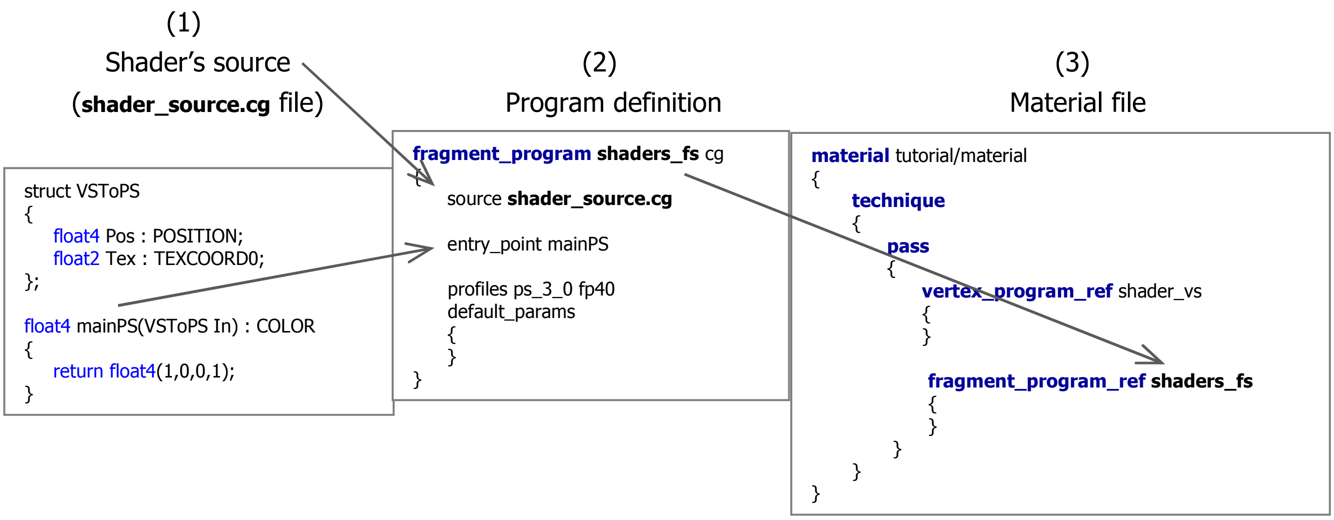


Figure 30: Ogre3D's shading system

6.2 Material File

The D'Fusion Exporters modify the material files to account for the use of tiShaders. You will however find it useful to understand which standard material parameters become obsolete when exporting with tiShaders, and how are tiShaders actually referenced in the material files.

This information can be useful if you want for example to reference your custom shaders by manually editing your exported material files, or when using shaders with particles and overlays.

6.2.1 Material Example with tiShaders

This is an example of material exported with tiShaders:

```
material alien/alien_eye_teeth_mat
{
  technique
  {
    pass
    {
      ambient 1 1 1 1
      diffuse 1 1 1 1
      specular 0.5 0.5 0.5 89.6
      emissive 0 0 0
```

(continue on next page)

```
        // vertex (skinning) shader
        vertex_program_ref tiShader/vertex_phong_skinning.vs
        {
          //input parameters for the program
        }

        // pixel shader (uses texture unit below)
        fragment_program_ref tiShader/phong_diffuse.ps
        {
          //input parameters for the program
        }

        // optional, texture unit
        texture_unit
        {
          texture Alien_6.png
        }

        // optional, texture unit for normal map
        texture_unit
        {
          texture normal.jpg
        }

      }
    }
  }
```

Note the use of a vertex and a fragment shader.

6.2.2 Pass Attributes

The export with tiShaders implements Phong and Gouraud shadings.

With Gouraud shading, lighting occurs at vertex level, then the texture is modulated with this result. In terms of visual result, this is equivalent to a standard rendering, and closer to OpenGL ES 1.

Phong shading is physically more accurate, leading to a different visual behavior: a unique diffuse texture is used. Ambient, diffuse specular and emissive attributes are thus interpreted in a slightly different way than before: diffuse is applied to the diffuse texture according Phong shading, then ambient, specular and emissive are applied to the global result.
Without shaders (Direct3D9, OpenGL and OpenGL ES 1.X) the result of the ambient + diffuse + specular + emissive is applied to the texture.

The following pass attributes have not been implemented, take a constant value or have a limited value:

- `alpha_rejection`: not taken into account by OpenGL ES 2 only, value = `none`
- `fog_override`: not taken into account, value = `none`
- `lighting`: not taken into account, value = `ON`
- `illumination_stage` and `shading` not taken into account
- `max_lights` not taken into account, value = `1`: 1 light maximum is implemented in tiShaders, this light can be of Point type only
- `start_light`: not supported

Stroked-through, non-supported attributes:

```
pass
     {
          ambient 1.0 1.0 1.0
          diffuse 1.0 0.5 0.5 1.0
          specular 1.0 1.0 1.0 12.5
          emissive 1.0 0.0 0.0

          scene_blend
          separate_scene_blend
          scene_blend_op
          separate_scene_blend_op
          depth_check
          depth_write
          depth_func
          depth_bias
          iteration_depth_bias
          alpha_rejection
          alpha_to_coverage
          light_scissor
          light_clip_planes
          illumination_stage
          transparent_sorting
          normalise_normals
          cull_hardware
          cull_software
          lighting
          shading
          polygon_mode
          polygon_mode_overrideable
          fog_override
          colour_write
          max_lights
          start_light
          iteration
          point_size
          point_sprites
          point_size_attenuation
          point_size_min
          point_size_max
}
```

6.2.3 Texture Attributes

With shaders, you can use textures for diffuse, normal and cubic environment maps. You can combine these textures to obtain the expected result.

The following texture attributes have not been implemented, take a constant value or have a limited value:

- `tex_coord_set`: not taken into account. The first set of texture coordinates is used with diffuse and normal mapping
- `colour_op`: not taken into account
- `colour_op_ex`: not taken into account
- `colour_op_multipass_fallback`: not taken into account

Stroked-through, non-supported attributes:

```
// optional, texture unit for diffuse lighting
    texture_unit
    {
        texture image_diffuse_lighting.jpg
        texture_alias
        anim_texture
        cubic_texture
        ~~tex_coord_set~~
        tex_address_mode
        tex_border_colour
        filtering
        max_anisotropy
        mipmap_bias
        ~~colour_op~~
        ~~colour_op_ex~~
        ~~colour_op_multipass_fallback~~
        ~~alpha_op_ex~~
        env_map
        scroll
        scroll_anim
        rotate
        rotate_anim
        scale
        wave_xform
        transform
        binding_type
        content_type
    }
```

6.3 Particles and Overlays

Particles and Overlays material files are not exported by the D'Fusion Exporters, which means that the use of tiShaders is not carried out automatically.

To use shaders with your particles and overlays, you must manually edit their corresponding material files to reference the appropriate shaders.

The use of shaders for your particles and overlays offers many advantages, including the use of rather advanced effects like environment maps.

Below you will find the description of a particle and an overlay material file using tiShaders.

6.3.1 Overlay

With overlays, the two tiShaders (one vertex, one fragment) to use are:

- tiShader/vertex_particle.vs: takes account of vertex position
- tiShader/TextureReplace.ps: to be used with overlays only: there is no lighting; the texture is directly applied as such.

6.3.1.1 Overlay material with tiShaders

The following code provides an example of an overlay material file using tiShaders. Compare with example 6.3.1.2 below "Overlay material file without shader".

```
material MaterialWithShaders
{
	technique
	{
		pass
		{
			scene_blend alpha_blend
			vertex_program_ref tiShader/vertex_particle.vs
			{
			}
			fragment_program_ref tiShader/TextureReplace.ps
			{
			}
			texture_unit
			{
				texture helloImAnOverlay.png
			}
		}
	}
}
```

6.3.1.2 Overlay material file without shader

```
material MaterialNoShaders
{
	technique
	{
		pass
		{
			scene_blend alpha_blend

			texture_unit
			{
				texture helloImAnOverlay.png
			}
		}
	}
}
```

6.3.2 Particles

With particles, the two tiShaders (one vertex, one fragment) to use are:

- tiShader/vertex_particle.vs: takes account of vertex position
- tiShader/particle_vertexColour.ps: takes account of vertex color (works both with particles and overlays).

The example below correspond to the material file of smoke particles.

6.3.2.1 Smoke particles with tiShaders

```
material Material/Smoke
{
      technique
      {
         pass
         {
               lighting off
               scene_blend alpha_blend
               depth_write off

               vertex_program_ref tiShader/vertex_particle.vs
                {
                }
                fragment_program_ref tiShader/particles_vertexColour.ps
                {
                }
               texture_unit
               {
                    texture smoke.png
                    tex_address_mode clamp
               }
         }
      }
}
```

6.4 tiShaders by Name

6.4.1 Vertex tiShaders

Vertex tiShader programs support:

- Vertex (hardware) skinning animations (limited to 39 bones per .mesh, 4 bones per vertex – see section 3.1.2)
- Specular highlights
- Normal (Phong only)
- 1 pointlight (see section 3.1.3)

Specific vertex tiShaders have been developed for particles and overlays (see section 6.3).
The following table lists Vertex tiShaders and implemented effects:

Shader name	Effect						
	Shading	Normal	Parity Bit	Hardware Skinning	Specular	Particles	Overlays
tiShader/vertex_phong.vs	Phong						
tiShader/vertex_phong_normal.vs	Phong	+					
tiShader/vertex_phong_normal_tangentW.vs	Phong		+				
tiShader/vertex_phong_skinning.vs	Phong			+			
tiShader/vertex_phong_skinning_normal.vs	Phong	+		+			
tiShader/vertex_phong_skinning_normal_tangentW.vs	Phong		+	+			
tiShader/vertex_gouraud.vs	Gouraud				+		
tiShader/vertex_gouraud_skinning.vs	Gouraud			+	+		
tiShader/vertex_gouraud_nospecular.vs	Gouraud						
tiShader/vertex_gouraud_nospecular_skinning.vs	Gouraud			+			
tiShader/vertex_particle.vs						+	+

Figure 31: Vertex tiShaders

6.4.2 Fragment tiShaders

Fragment tiShader programs support:

- Phong/Gouraud shading
- Diffuse maps
- Normal maps (with the limitations explained in 3.1.4)
- Environment maps (with the limitation explained in 3.1.5)
- Specular highlights
- Alpha-blend
- 1 pointlight (the one closest to the object is used)

Specific fragment tiShaders have been developed for particles and overlays (see section 6.3).
The following tables lists Fragment tiShaders and implemented effects:

	Effect							
Phong Shaders	Shading	Diffuse map	Normal map	EnvCubic map	No Alpha Blend	Specular	Pointlight	Spotlight* (*pointlight or spotlight, not both)
tiShader/phong.ps	Phong					+	+	
tiShader/phong_diffuse.ps	Phong	+				+	+	
tiShader/phong_diffuse_normal.ps	Phong	+	+			+	+	
tiShader/phong_normal.ps	Phong		+			+	+	
tiShader/phong_envcubic.ps	Phong			+		+	+	
tiShader/phong_envcubic_separate.ps	Phong			+	+	+	+	
tiShader/phong_diffuse_envcubic.ps	Phong	+		+		+	+	
tiShader/phong_diffuse_envcubic_separate.ps	Phong	+		+	+	+	+	
tiShader/phong_diffuse_normal_envcubic.ps	Phong	+	+	+		+	+	
tiShader/phong_diffuse_normal_envcubic_separate.ps	Phong	+	+	+	+	+	+	
tiShader/phong_normal_envcubic.ps	Phong		+	+		+	+	
tiShader/phong_normal_envcubic_separate.ps	Phong		+	+	+	+	+	
tiShader/phong_nospecular.ps	Phong							
tiShader/phong_nospecular_diffuse.ps	Phong	+					+	
tiShader/phong_nospecular_diffuse_normal.ps	Phong	+	+				+	
tiShader/phong_nospecular_normal.ps	Phong		+				+	
tiShader/phong_nospecular_envcubic.ps	Phong			+			+	
tiShader/phong_nospecular_envcubic_separate.ps	Phong			+	+		+	
tiShader/phong_nospecular_diffuse_envcubic.ps	Phong	+		+			+	
tiShader/phong_nospecular_diffuse_envcubic_separate.ps	Phong	+		+	+		+	
tiShader/phong_nospecular_diffuse_normal_envcubic.ps	Phong	+	+	+			+	
tiShader/phong_nospecular_diffuse_normal_envcubic_separate.ps	Phong	+	+	+	+		+	
tiShader/phong_nospecular_normal_envcubic.ps	Phong		+	+			+	
tiShader/phong_nospecular_normal_envcubic_separate.ps	Phong		+	+	+		+	

tiShader/phong_PoinSpot.ps	Phong					+	+	+
tiShader/phong_PoinSpot_diffuse.ps	Phong	+				+	+	+
tiShader/phong_PoinSpot_diffuse_normal.ps	Phong	+	+			+	+	+
tiShader/phong_PoinSpot_normal.ps	Phong		+			+	+	+
tiShader/phong_PoinSpot_envcubic.ps	Phong			+		+	+	+
tiShader/phong_PoinSpot_diffuse_envcubic.ps	Phong	+		+		+	+	+
tiShader/phong_PoinSpot_diffuse_envcubic_separate.ps	Phong	+		+	+	+	+	+
tiShader/phong_PoinSpot_diffuse_normal_envcubic.ps	Phong	+	+	+		+	+	+
tiShader/phong_PoinSpot_diffuse_normal_envcubic_separate.ps	Phong	+	+	+	+	+	+	+
tiShader/phong_PoinSpot_normal_envcubic.ps	Phong		+	+		+	+	+
tiShader/phong_PoinSpot_normal_envcubic_separate.ps	Phong		+	+	+	+	+	+
tiShader/phong_PoinSpot_nospecular.ps	Phong						+	+
tiShader/phong_PoinSpot_nospecular_diffuse.ps	Phong	+					+	+
tiShader/phong_PoinSpot_nospecular_diffuse_normal.ps	Phong	+	+				+	+
tiShader/phong_PoinSpot_nospecular_normal.ps	Phong		+				+	+
tiShader/phong_PoinSpot_nospecular_envcubic.ps	Phong			+			+	+
tiShader/phong_PoinSpot_nospecular_envcubic_separate.ps	Phong			+	+		+	+
tiShader/phong_PoinSpot_nospecular_diffuse_envcubic.ps	Phong	+		+			+	+
tiShader/phong_PoinSpot_nospecular_diffuse_envcubic_separate.ps	Phong	+		+	+		+	+
tiShader/phong_PoinSpot_nospecular_diffuse_normal_envcubic.ps	Phong	+	+	+			+	+
tiShader/phong_PoinSpot_nospecular_diffuse_normal_envcubic_separate.ps	Phong	+	+	+	+		+	+
tiShader/phong_PoinSpot_nospecular_normal_envcubic.ps	Phong		+	+			+	+
tiShader/phong_PoinSpot_nospecular_normal_envcubic_separate.ps	Phong		+	+	+		+	+

Figure 32: Phong Fragment tiShaders

Gouraud Shaders	Effect							
	Shading	Diffuse map	Normal map	EnvCubic map	No Alpha Blend	Specular	Pointlight	Spotlight* (*pointlight or spotlight, not both)
tiShader/gouraud.ps	Gouraud					+		
tiShader/gouraud_diffuse.ps	Gouraud	+				+		
tiShader/gouraud_diffuse_envcubic.ps	Gouraud	+		+		+		
tiShader/gouraud_diffuse_envcubic_separate.ps	Gouraud	+		+	+	+		
tiShader/gouraud_envcubic.ps	Gouraud			+		+		
tiShader/gouraud_envcubic_separate.ps	Gouraud			+	+	+		
tiShader/gouraud_nospecular.ps	Gouraud							
tiShader/gouraud_nospecular_diffuse.ps	Gouraud	+						
tiShader/gouraud_nospecular_diffuse_envcubic.ps	Gouraud	+		+				
tiShader/gouraud_nospecular_diffuse_envcubic_separate.ps	Gouraud	+		+	+			
tiShader/gouraud_nospecular_envcubic.ps	Gouraud			+				
tiShader/gouraud_nospecular_envcubic_separate.ps	Gouraud			+	+			

Figure 33: Gouraud Fragment tiShaders

Particles and Overlays	Effect	
	Particles	Overlays
tiShader/TextureReplace.ps		+
tiShader/particle_vertexColour.ps	+	

Figure 34: Particles and Overlays Fragment tiShaders

6.5 tiShaders Mobile Relative Performance Benchmarks

The following table illustrates the performances obtained from several tiShader combinations for different devices.

Performances are given relative to a given mobile device.

For example, regarding rendering FPS, an index of performance of a 100 has been given to the Galaxy S II. For a given shader combination, a device (other than the Galaxy S II) with an index of 90 is less performing than a device (other than the Galaxy S II) with an index of 120, relative to the performance obtained for the Galaxy S II device.

These relative benchmark are intended to be used to evaluate, for a given Media and a given device,

the possible gain/loss in performances in changing its shaders,

the probable performances of the media on an other device.

Please note that the following benchmarks have been passed in using very basic and typical tests.

Mobile device	Samsung Galaxy S II	Samsung Galaxy Tab	HTC Nexus One	Apple iPhone 4S	Apple iPad3
Graphic chipset	Mali-400 MP	Power VR SGX 540	Adreno 200	Power VR SGX 543	Power VR SGX 543
	Resolution 480x800 EGL context = 5,6,5,0 FSAA=4	Resolution 1280x720 EGL context = 5,6,5,0 FSAA=4	Resolution 480x800 EGL context = 5,6,5,0 FSAA=4	Resolution 640x960 Activated retina display	Resolution 1536x2048 Activated retina display
Rendering Fps performance (% relatively to GxS II)					
Gouraud shading + Diffuse map	100,00	89,83	49,15	100,68	99,56
Gouraud shading + Diffuse map + Hardware skinning	100,00	91,53	77,97	100,51	100,42
Phong shading + Diffuse map	100,00	50,85	25,59	86,64	68,43
Phong shading + Diffuse map + Hardware skinning	98,31	49,15	22,03	99,90	70,23
Percentage of free CPU	**relatively to GxS II**			**relatively to iPhone 4S**	
Gouraud shading + Diffuse map	100,00	75,81	50,00	100,00	102,31
Gouraud shading + Diffuse map + Hardware skinning	120,97	108,06	103,23	159,09	179,74
Phong shading + Diffuse map	100,00	95,16	80,65	95,30	68,70
Phong shading + Diffuse map + Hardware skinning	120,97	106,45	122,58	130,51	85,77

Figure 355: tiShaders relative mobile performances (3D Mesh with 3D Skinning)

Mobile Device	Samsung Galaxy S II	Samsung Galaxy Tab	HTC Nexus One	Apple iPhone 4S	Apple iPad3
Graphic chipset	Mali-400 MP	Power VR 540	Adreno 200	Power VR SGX 543	Power VR SGX 543
	Resolution 480x800 EGL context = 5,6,5,0 FSAA=4	Resolution 1280x720 EGL context = 5,6,5,0 FSAA=4	Resolution 480x800 EGL context = 5,6,5,0 FSAA=4	Resolution 640x960 Activated retina display	Resolution 1536x2048 Activated retina display
Rendering FPS performance (% relatively to GxS II)					
Gouraud shading	100,00	91,53	66,10	100,85	99,63
Gouraud shading + Diffuse map	99,83	84,75	64,41	100,68	99,63
Gouraud shading + Diffuse map + Cubic env. map	99,83	81,36	57,63	99,32	99,48
Phong shading	100,00	38,98	27,12	100,39	82,06
Phong shading + Diffuse map	96,61	5,08	23,73	100,78	73,04
Phong shading + Diffuse map + Cubic env. map	96,61	28,81	20,34	100,00	49,16
Phong shading + Diffuse map + Cubic env. map + Normal map	66,10	3,39	14,58	84,39	37,63
Percentage of free CPU	**relatively to GxS II**			**relatively to iPhone 4S**	
Gouraud shading	100,00	91,46	78,05	100,0	75,9
Gouraud shading + Diffuse map	98,78	90,24	80,49	97,7	65,8
Gouraud shading + Diffuse map + Cubic env. map	97,56	89,02	79,27	70,2	63,4
Phong shading	96,34	93,90	102,44	112,6	79,4
Phong shading + Diffuse map	97,56	113,41	100,00	160,4	96,8
Phong shading + Diffuse map + Cubic env. map	97,56	90,24	101,22	114,9	53,0
Phong shading + Diffuse map + Cubic env. map + Normal map	97,56	109,76	101,22	63,7	31,0

Figure 366: tiShaders relative mobile performances (3D Mesh without 3D Skinning)